NPSCS-92-002

NAVAL POSTGRADUATE SCHOOL
Monterey, California

RULE BASED CONTROL FOR MISSION EXECUTION FOR AN AUTONOMOUS UNDERWATER VEHICLE

by

Yuh-jeng Lee and Paul Wilkinson

February 1992

Prepared for:

Naval Postgraduate School
Monterey, California 93943

NAVAL POSTGRADUATE SCHOOL

Monterey, California

REAR ADMIRAL R. W. WEST, JR.
Superintendent

HARRISON SHULL
Provost

This report was prepared for and funded by the Naval Postgraduate School.

This report was prepared by:

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSCS-92-002	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) Naval Postgraduat School Monterey, CA 93943-5100
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER OM & N Direct Funding
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Rule Based Control for Mission Execution for an Autonomous Underwater Vehicle

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1992, February, 6	29

16. SUPPLEMENTARY NOTATION
The views expressed in this tech report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Expert Systems Autonomous Underwater Vehicle
Rule Based Control, Mission Execution

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report describes a rule based system that has been designed to oversee the maneuvering of the autonomous underwater vehicle, AUV II, that has been constructed at the Naval Postgraduate School. The system will monitor the progress from the AUV launch point to a goal area and back to the origin. It is able to make informed decisions about the mission, taking into account the navigational path, the vehicle subsystems health, the sea environment, and the specific mission profile which is downloaded from an off-board mission planner. Heuristics for maneuvering, avoidance of uncharted obstacles, waypoint navigation, and reaction to emergencies - essentially the expert knowledge of a submarine captain - have been coded using the expert system shell CLIPS. The design of a high level control software architecture for AUV II and the development of the domain specific knowledge for AUV operation are discussed in detail. Simulation results showed that the system is capable of reacting to various adverse situations in a timely manner.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Yuh-jeng Lee	22b. TELEPHONE (Include Area Code) (408) 646-2361 / 22c. OFFICE SYMBOL CS/Le

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

Rule Based Control for Mission Execution for an Autonomous Underwater Vehicle

Yuh-jeng Lee
Paul Wilkinson

Computer Science Department
Naval Postgraduate School
Monterey CA 93943

Abstract

This paper describes a rule based system that has been designed to oversee the maneuvering of the autonomous underwater vehicle, AUV II, that has been constructed at the Naval Postgraduate School. The system will monitor the progress from the AUV launch point to a goal area and back to the origin. It is able to make informed decisions about the mission, taking into account the navigational path, the vehicle subsystems health, the sea environment, and the specific mission profile which is downloaded from an offboard mission planner. Heuristics for maneuvering, avoidance of uncharted obstacles, waypoint navigation, and reaction to emergencies – essentially the expert knowledge of a submarine captain – have been coded using the expert system shell CLIPS. The design of a high level control software architecture for AUV II and the development of the domain specific knowledge for AUV operation are discussed in detail. Simulation results showed that the system is capable of reacting to various adverse situations in a timely manner.

1 Introduction

For decades, the development of autonomous underwater vehicles (AUV's) has been an ambition for industries and the government alike. Only recently, however, have practical AUV's appeared to be reality. Several marine autonomous and remotely-piloted vehicles are currently in use for such diverse functions as underwater inspection [9], offshore oil exploitation [4], and hydrography [13]. The practical advantage of low-risk to human operators coupled with the potential ability to operate at over-the-horizon distances make autonomous underwater vehicles highly desirable for many subsea activities and operations.

The Naval Postgraduate School has been investigating AUV technologies involving vehicle dynamics and control, artificial intelligence, robotics, and computer architectures. An experimental testbed, AUV II, has been designed, fabricated, and successfully

launched on site [7]. As noted in [15], the technologies that are essential to AUV success include navigation, composite hull materials, guidance, energy source, propulsion, communication links, and signal processing as well as the specific mission packages. To make an AUV truly adaptive and survivable, however, an advanced decision making capability is also needed. Our current efforts attempt to take the development of an intelligent control system for AUV II into the next increment of evolution, beyond the primitive closed-loop control. The project focuses on software modules in two different levels: the intermediate level modules, such as pattern recognition and navigation, and high level modules, such as mission planning/replanning and mission execution.

This paper describes the design and development results of a mission executor which is responsible for high level control of AUV II. The purpose is to provide a control system for the vehicle so that it will adequately emulate the human-machine interaction that regularly takes place on manned submersibles.

1.1 The Need for a Mission Execution Expert System

One general software architecture for AUV is a top-down flow of control ranging from strategic level control through tactical level to hardware interface level. Higher levels of abstraction perform some of the activities (some time-sensitive) which require measured decision-making.

The control structure of AUV II has undergone an evolutionary development. Initially, it was essentially low level closed-loop control. Incremental enhancements to the AUV functionalities necessitated the incorporation of an expert system to integrate and coordinate intelligent activities such as system monitoring, waypoint following, and obstacle avoidance. As the autonomous vehicle enters the real underwater environment, it must perform independently all kinds of tasks required to complete its mission. Although routine situations can be handled in an algorithmic manner, many incidental problems need to be resolved relying on heuristics. This type of problem solving requires high level reasoning and decision making, often in real-time, and in an environment of uncertainty or incomplete knowledge. The primary goal of the *Mission Executor* was to design

heuristics that make it possible for the system to deal with extensions of well-known problems.

1.2 Requirements of a Mission Executor

The *Mission Executor*, in the broadest sense, must be able to safely control movement between a mission starting point and a mission goal. In doing so, it must operate between three models: that of the vehicle world, the environmental world, and the mission world. To supervise the vehicle world implies that the *Mission Executor* must monitor and control vehicle "health" such as battery state, internal system pressure, and temperature. It must also be able to respond to a deteriorating condition of the vehicle sonar, navigation system, or guidance systems. The loss of a major onboard instrument, such as the sonar or navigation system, would probably be catastrophic in many cases and would result in a mission degradation in the least serious case. The *Mission Executor* must supervise the subsystem recovery procedure or make decisions that can circumvent the problem. Should that fail it must make a strategic level decision to abort the mission.

Control of the vehicle in the context of the environmental world means reaction to topological features such as undersea terrain and obstacles (both moving and non-moving), a significant change in atmospheric conditions, or any external threat which would physically hinder the vehicle from making the transit to the goal point.

Monitoring of the mission world entails recognizing transition points between handling normal transit and beginning a special mission profile. Possible speed and depth changes, special requirements for inshore navigation, and deployment of any equipment must be considered. Most important, the mission priority must be readjusted for vehicle survival and reusability. Heuristics for this must be incorporated in the software.

1.3 Organization of the Paper

Section 2 is a survey of related work on AUV control systems and related technology. Section 3 describes information processing onboard the AUV II. It details the interactions between various modules of the AUV II software architecture. Section 4 presents the

design of the *Mission Executor*. Section 5 provides a description of the implementation of the *Mission Executor*. Issues on the proper combination of rules and objects, the role of uncertainty and truth maintenance are also discussed in the context of AUV operations. Section 6 presents simulation results. Section 7 outlines contributions, conclusions, and extensions for further work.

2 Related Work

Layered control architecture [3] is behavior-oriented, using the subsumption approach: low-order behaviors are first installed and verified in the testbed, and when satisfactory performance is achieved, the next level of complex behaviors is then added. The lower level is subsumed by the higher level. Other studies that employ some form of the layered architecture include: (1) state configured layered control [1] which addresses the issue of mission specific behavior coordination; (2) the ARCS underwater vehicles [19], which also incorporate rule based heuristics and learning through reflexive behaviors, logical behaviors, and learned behaviors; (3) the TASC/NUSC architecture [17] which combines aspects of real-time layering, functional decomposition and subsumption; and (4) KB/EAVE [2], which is knowledge-based AUV software that accomplishes its functions through functional layering.

In the hierarchical structure approach, the software is usually divided hierarchically into interrelated levels of functional units such as the planner, navigator, pilot, and actuator/controller [8]. Each level has its own separate sensor bank for perception, a map for world model reasoning, and a reporter for intelligent control. The functional unit itself has its own database, rule base, and evaluator. Software for Autonomous Control of ROV90 [16] is also a hierarchical system in which a monitoring unit, the Watchdog, supervises several lower level modules that perform the functions of mission sequencing, navigation, vehicle control, and error checking.

The Exception Handling Model [14], constructed as a procedural expert system, functions as a mission executor for an industrial robot. It attempts to achieve the planned behavior and provides a series of prioritized strategies for recovery. The strategies are

encoded in heuristics around the general functions of monitoring, diagnosis, and response in a loose hierarchy.

3 Onboard Information Processing

While the AUV II software contains over a dozen modules [7], only four of them directly interact with the *Mission Executor*. The interface between these modules and the *Mission Executor* is depicted in Figure 1. Each of the modules and the data transmitted between the modules are discussed in the following sections.

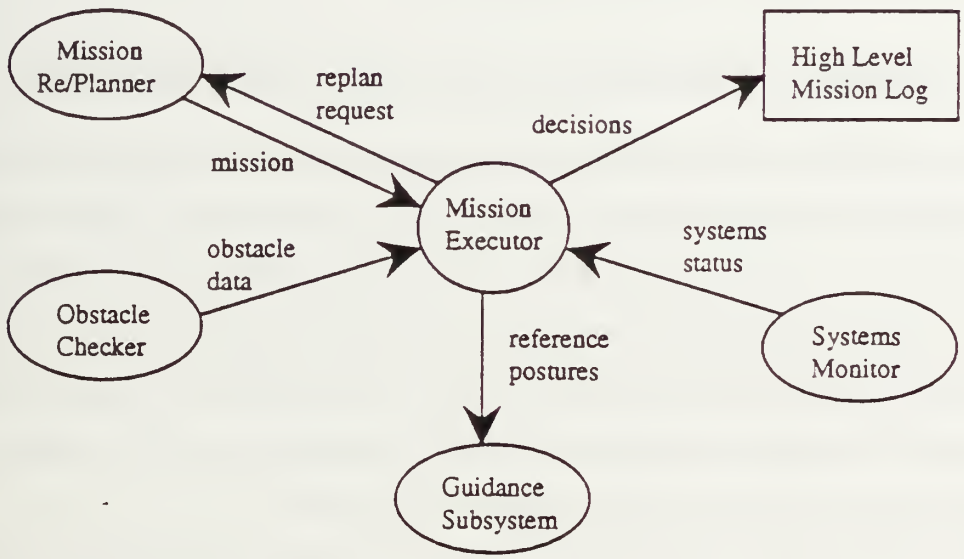

Figure 1: The Interface

3.1 Downloading Mission Profile from the Mission Planner

Given mission requirements and relevant environment information, the offboard *Mission Planner* provides AUV II with initial instructions to carry out the mission. These include the best three dimensional operation path to the goal, time requirements, and special path constraints [11]. The *Mission Executor*'s most important functions in a normal transit are to receive waypoints and command data (denoted as a path) from the *Mission Planner*, to interpret the movements, to convert path postures to reference postures, and to properly sequence and monitor the movements. The other important functions which

the *Mission Executor* carries out are generally exception-handling relative to normal operations.

The *Mission Executor*'s role in interacting with the *Mission Planner* should not be categorized as simply a conversion unit serving a high-level planner. The *Mission Executor* must reason about waypoints and their associated speeds. If the original commanded speed for a particular waypoint is not feasible due to an unplanned deviation from course, then the *Mission Executor* must call the *Navigation* module for an updated speed to reach the goal on time. It may also request the onboard *Mission RePlanner* to replan the mission should the original path be no longer valid.

3.2 Updating World Model with the Obstacle Checker

Conceptually, the *Obstacle Checker* has the responsibility of processing packaged sensor data transmitted from the sonar module and relating it to specific obstacles. Both the type of obstacles (moving or stationary) and the avoidance maneuver (decrease-speed, increase-speed, dive, ascend, or stop) are determined and passed to the *Mission Executor*. The data passed include an obstacle alert-and-direction flag followed by a template containing the following information: obstacle identification, relative distance, relative orientation, time, movement, and parameters of movement.

The direction flag is sent merely to alert the *Mission Executor* for a real-time report. Receipt of the template data allows the *Executor* to call the *RePlanner* with the information. The *Executor* also flags *Guidance* to be ready for imminent receipt of revised reference postures for the new path-to-goal referenced to the current geographical position. A low-level reflexive response can also be passed directly to the guidance controller, thereby bypassing the *Mission Executor* in the case of an unplanned obstacle close aboard.

3.3 Requesting Replanning

The *RePlanner*, a knowledge based path planner that uses an optimized real-time A* search [10], attempts to plan a new path-to-goal based on knowledge of the goal state,

the current geographical location, and special path constraints passed by the *Executor*. It operates in four dimensions: three standard cartesian dimensions and a fourth dimension of heading or azimuth [12]. The *RePlanner* receives periodic updates from the environmental database, allowing it to replan the new route from any specified origin.

The *RePlanner* is alerted to the need to replan by a function call from the *Executor*. A flag and the coordinates of the current location are transferred to the *RePlanner*. It constructs a new plan in the same manner as the *Planner* does, using *a priori* knowledge of the environment. A file of new waypoints is returned to the *Mission Executor* when replanning is completed.

3.4 Guarding Vehicle Health with Systems Monitor

The vehicle's internal world is modeled as a set of sensor objects which measures the subsystem components, including power sources such as an array of batteries for subsystem power and propulsion support, control system indicators for rudders, planes and propellers, sonar power status indicators for four onboard sonars, onboard computer temperature sensors, navigation instrument fault sensors, and power sources for environmental sensors. These have default guard-line and red-line ranges which, when violated, cause an alarm to be sent to the decision making levels. An automated turn-key operation is first generated which attempts to recover from an equipment failure or impending failure by bringing a redundant system on-line, if such redundancy is provided. If the equipment is critical, it may degrade the vehicle condition or even abort the mission.

3.5 The Guidance Subsystem

The end results of the *Mission Executor* functions must be a series of reference postures and commands to the *Guidance* subsystem. *Guidance* is an intermediate level function which has an algorithmic reasoning system. It converts high level decisions and reference postures to low level commanded postures for the *Autopilot* module. A function call within the rules of the *Mission Executor* generates an alert to the *Guidance* module to prepare for receipt of data and commands.

4 Design of the Mission Executor

This section describes the design of the *Mission Executor*, interface limitations, and the software constructs used. This design is intended to cover most situations of AUV operations.

4.1 Reasoning about Several Worlds

The *Mission Executor* attempts to serve the role of high level director while integrating decisions on the basis of input from the vehicle internal systems, the external environment, and the mission plan. It continuously assesses whether a mission can be carried out successfully, the ultimate goal. Decisions are modeled heuristically rather than in a strictly algorithmic fashion, based primarily on the status of low level events. In other words, low level events drive broader decisions. The requirement to model this lends itself naturally to a hierarchical design, but one that is priority-situation based.

Basic AUV guidance or control systems are closed-loop and are adequate to handle routine maneuvering. The *Mission Executor* exists mainly to deal with exceptions to normal maneuvering which cannot be processed in a strictly algorithmic manner. Its reasoning results in interrupt commands to the *Guidance* subsystem (which in turn controls the *Autopilot*). If there are no deviations from the track caused by any of the three worlds that AUV must deal with, then the *Mission Executor* merely fulfills the role of sequencer of data.

Although not all experiential knowledge may be encoded in rules, there is reason to believe that AUV missions can be bounded. Some previous research has suggested that AUV behaviors can in fact be standardized. For example, typical situations in which an AUV might find itself have been identified in [18] in which the "Generalized Problem vs Contingency Alternatives Matrix" provides a set of possible problems and alternative actions an AUV can take to overcome the problems. Problems are classified into three categories: mission, environment, and internal failures. These have one-to-one correspondence with the three worlds that the *Mission Executor* tries to model.

This view of high level control as essentially handling exceptions to normal transit and operations is embodied in the *Mission Executor*. Some of the implications of the matrix merit serious consideration while others are simply beyond the scope of current technology. For example, vehicle self-repair is highly unlikely in a mechanical failure situation unless this term refers only to equipment which has a redundant system or power source available.

4.2 The Structure

The *Mission Executor* consists of a knowledge base of rules, facts, and objects. Figure 2 shows the overall structure and data flows of the *Mission Executor*. A set of rules exists for the *Mission Executive* and for each functional (i.e., situational) area: maneuvering, navigation, subsystem-monitoring, environmental hazard, and specialized mission. The rules interact with the object base and cache of facts to produce guidance commands.

The *Mission Executor* was designed based on the overall state of the mission existing in one of three forms: continue_unrestricted, continue_with_restrictions, or abort_mission. If no deviations occur during the course of the mission, the mission status remains at its default status, continue_unrestricted. It views each functional area in two levels: critical and failure. The critical level indicates that a functional area has suffered some sort of restrictive, non-catastrophic loss of capability. This can be on the order of loss of non-mission essential equipment or a temporary maneuvering restriction such as obstacle avoidance which takes it away from its principal direction of travel. This results in the mission status of continue_with_restrictions. The mission restriction category can later be lifted if the vehicle recovers in ample time. If not, the mission restriction remains, or the overall mission status worsens to abort_mission. The failure level indicates that the functional area has suffered a major loss of capability such as loss of mission-essential equipment or inability to maneuver. This essentially results in the status of abort_mission.

Each of the functional areas has a hierarchy among its rules. A functional assessor exists at the top of each rule base to cache knowledge about the functional area. The

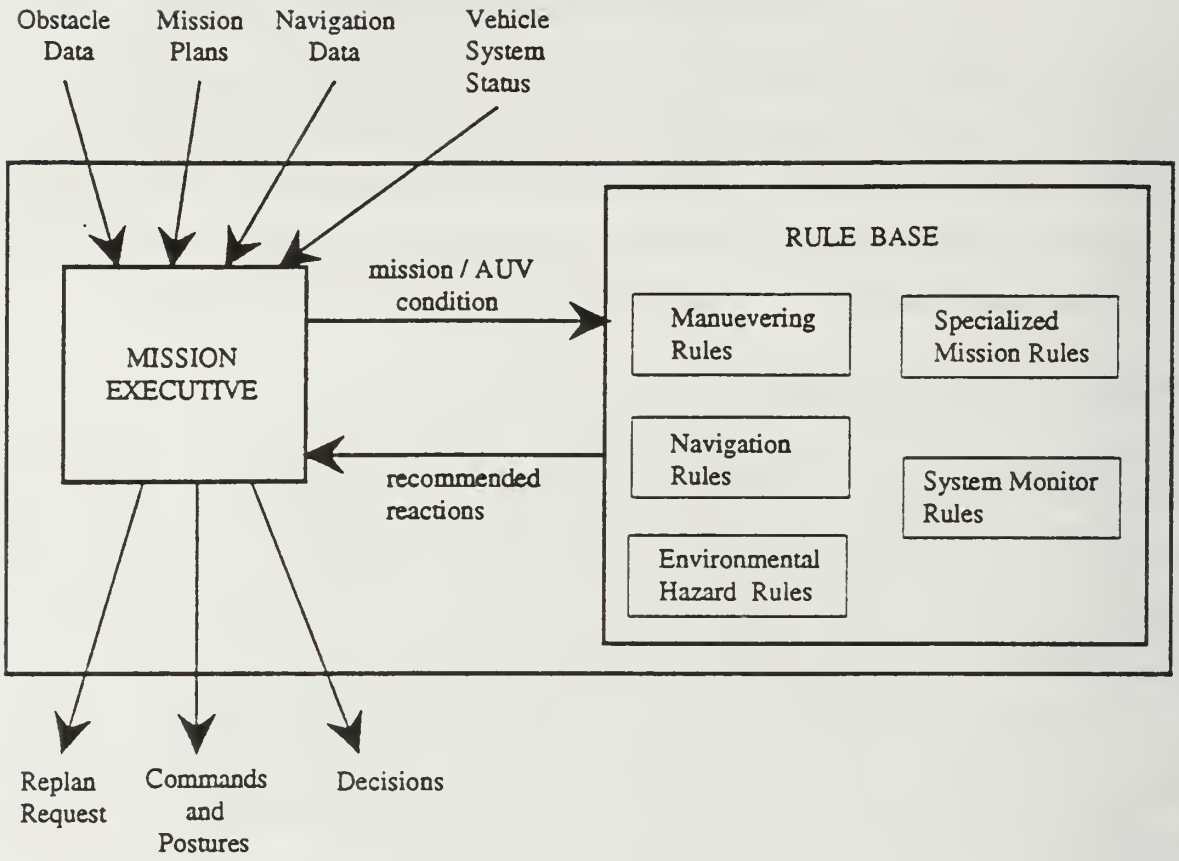

Figure 2: The structure of the *Mission Executor*

functional area information is then passed to the *Mission Executive* and causes top level decision rules to be fired.

4.2.1 The Mission Executive

The *Mission Executive* performs the highest level of reasoning in the *Mission Executor* and consists of three major components: Mission_Interpreter, Mission_Monitor, and Overall_Mission_Assessor. The Mission_Interpreter receives mission plans from the offboard *Mission Planner* and provides instructions to carry out the mission. The Mission_Monitor is in charge of routine background functions such as the sequencing of the mission timer and the continuous loop that queries the environmental sensors and the internal vehicle subsystems. The Overall_Mission_Assessor is responsible mainly for tabulating the results

sent by the functional area assessors. It is insulated from details of the reports by the functional area supervisors.

If the AUV enters the status of abort_mission, the *Mission Executive* will request the vehicle path to be replanned for a pre-planned rendezvous point. It may be the origin of the mission or an intermediate point which facilitates recovery by the launching platform. The status of continue_with_restrictions allows the vehicle to try to recover from its maneuvering, navigation, or equipment restrictions.

The rules in the *Mission Executive* are given the highest priority for execution. It is necessary to differentiate between a high-level, less frequent action and a lower-level, frequently performed action. This is because a situation (pattern match) which may cause an abort_mission or continue_with_restriction usually requires immediate or timely reaction and certainly takes precedence over a routine action such as a normal turn or depth-change. The emergency-action rule must be fired prior to other semantically lower-priority rules on the agenda. This heuristically models a submarine commander's "situational awareness" in an emergency. It might also be likened to the focus of attention approach, such as that modeled in [2].

Last, but not least, the *Mission Executive* must send reference postures as well as commands to the *Guidance* module. Many of the commands must initiate low level actions with real-time constraints while the assessment of a particular functional area status is in progress. The commands must be a series of well-understood actions which will place the vehicle in a safe configuration when a casualty occurs. Table 1 shows the commands currently implemented.

4.2.2 The Rule Base

The sequence of control in a rule-based system often contains a relatively high degree of non-determinism because of its declarative nature. While there are certain tasks which must be accomplished in procedural order, the *Mission Executor* is a system that reasons about situations which are normally beyond a closed-loop control system. Below we discuss the characteristics of major rules in each functional area.

Table 1: Possible commands to the *Guidance module*

Basic Maneuver	Order	Target of Order
turn	turn left	rudder
turn	turn right	rudder
depth change	ascend XX	planes
depth change	dive XX	planes
depth change	surface	planes
speed change	increase speed	drive motors
speed change	decrease speed	drive motors
speed change	stop	drive motors
speed change	hover	hover thrusters
XX = depth in inches or an indicated safe depth variable		

Maneuvering Rules Maneuvering rules cover several situations. First and foremost is obstacle avoidance. The highest priority rules cover emergency situations such as detection of an obstacle close aboard. Various orientations of the obstacle relative to the AUV's heading will prompt a right or left turn, an ascent or a full stop (drive motors stopped) or a combination of these. These are heuristic turning rules which can produce an effective gross avoidance for the AUV so that the *RePlanner* can then be invoked for further path refinement. Table 2, based on [5], shows the avoidance heuristics currently employed.

Table 2: Obstacle avoidance heuristics

Alert Flag	Turn	Depth Change
0XX0	—	—
0XX1	—	ascend
1101	left	ascend
1100	left	—
1011	right	ascend
1010	right	—
111X	stop	ascend

Alert Flag is a four-bit signal generated by the *Obstacle Checker*. They represent the information collected from the four sonars (forward, right, left, and bottom) installed on AUV II. For each bit, a 0 means that no obstacle is detected by that sonar, a 1 signals detection, an X can be either 1 or 0 (meaning the value is not important in determining maneuvering). For example, the alert flag 1010 (second line from bottom in the table) indicates that obstacles are detected by the forward and left sonars; therefore, the AUV should make a right turn to avoid collision.

Detection of an obstacle at the range of the sonar's limits is another function covered by the maneuvering rules. Because of the relatively limited distance of the AUV sonar, avoidance action must be taken early. An obstacle is initially checked for its potential to hazard the AUV, dependent on the obstacle's bearing drift and its relative bearing. This is recorded and a collective obstacle heuristic is triggered to determine whether a proportional amount of the obstacle will block the AUV to the left or right. A gross avoidance maneuver is then commanded to bring the AUV away from the obstacle and allow the *RePlanner* to plan the new avoidance path with appropriate mapping waypoints.

The procedures for creating an update to an obstacle are essentially the same. If the obstacle is still a hazard, then further avoidance and replanning are necessary. There is a chance that this will result in a significant deviation from the path and, eventually, a mission abort. On the other hand, if the obstacle is no longer a danger, then its collision danger is recorded as such and thus it is not considered in the collective obstacle assessment.

Other rules in the maneuvering functional area cover special depth-changing evolutions such as diving, ascents, and surfacing. The control systems have an inherently large influence on these special maneuvers. If a control system fails during one of these situation, that results in an automatically commanded maneuver to the *Guidance* to correct the altitude and level the vessel at a safe depth or change the speed at which the maneuver is proceeding. An improper obstacle clearance can also precipitate changing one of these special evolutions.

Navigation Rules The lower level action attribute of the configuration actually is instantiated within the Navigation rules upon the occasion of waypoint arrival. The rule that plays a significant part in the navigational aspect of high level control is the assessment of progress along the mission track. The rule makes a simple comparison of overall distance along the track with current location. It then orders a replan of the current track if the current speed and progress made are not compatible with reaching the goal area on time. A very simple energy-consideration function checks whether there is sufficient propulsive power to get to the goal.

Other navigation rules cover specially-monitored depths: both yellow-depth and red-depth. If the depth sonar indicates that the AUV has encountered a yellow depth area, the AUV calls the *Navigator* for a check of the required depth in that area. If the observed depth does not match the required depth, the *Guidance* is ordered to reverse course and the *RePlanner* is called. If a red-depth violation is indicated, the *Guidance* is called to reverse course.

System Monitor Rules Not only a large influence for its own functional area, the overall equipment status exerts a notable influence in other areas. Separate rules exist for each equipment area (sonar, control system, navigation instrument, environmental sensor, and special mission equipment) and the respective power source. A continuous monitoring rule polls each equipment area for equipments which are out of normal operating limits. These limits are normally parameters of sustenance such as potential in volts or power in watts. If a piece of mission essential equipment fails, it causes a failure in both the equipment status area and in the area with which it is associated. For example, loss of the diving-plane controls causes a maneuvering loss and a mission essential equipment loss. If an auxiliary power source exists for the equipment, it can be used in the event that the normal source fails. Similarly, equipment with redundancy has the capability to have its functions shifted to the alternate should it fail.

The Equipment Status Assessor awaits the results of equipment polling. If a piece of equipment fails, it will cause its classification rule to fire and the Equipment Status

Assessor will tabulate the results. If some mission-essential equipment or a sufficient number of non-mission-essential equipment fails, the equipment status area will suffer a major failure.

Environment Rules The Environment rules have a similar arrangement. The Environmental Assessor tabulates the number of sensors which have performance readings outside the limits. If it is essential equipment such as a pressure transducer, the loss will cause a functional area loss. If it is non-mission essential equipment, the loss will only cause a minor degradation to the environment functional area.

Special Mission Rules While basic AUV maneuvering control and navigation will be the primary focus for some time, incorporation of specialized missions will eventually become important. Specialized Mission rules have a different influence than the previous functional areas. Most of these rules do not take effect until the transition to a special mission configuration at the conclusion of the transit. The exception to this is a special mission area equipment failure. A functional mission area failure occurs if the special mission equipment fails. It is desirable to have an alternative to undertake a secondary mission if the primary mission cannot be fulfilled. The specialized rules are usually mission-dependent. There are no special mission rules in the present version of the *Mission Executor*.

5 Implementation

A prototype *Mission Executor* has been implemented using the expert system shell CLIPS version 5.0 [6], chosen for its wide availability (can be obtained at minimal cost), high portability (comes with source code written in C and can easily be modified), and flexibility (can easily be integrated with other programming languages or external systems). CLIPS was developed by NASA and has been used in the construction of both real-time and non-real-time systems. It is a forward-chaining, rule based software tool that provides

many desirable features including list processing, production system type of reasoning, and object oriented computing.

5.1 Main Algorithm

The vehicle reasoning system is initialized upon the download of the mission plan. This triggers the rule Mission_Timer, which continually binds the mission time to the current central processing unit (cpu) time. A timer flag is continually asserted in this rule and retracted in the timer manager rule. The timer manager continually asserts facts which trigger other polling rules. The initial state of mission_status is continue_unrestricted which will remain the same through the duration of the mission as long as no functional area becomes critical or experiences failure. Most of the rules in the *Mission Executor* are to handle missions which cannot remain in the ideal state due to a casualty or discrepancy in the mission, vehicle, or environmental worlds. The main algorithm can be described as follows:

```
download mission plan;
if vehicle_status = operational, then
   process mission_file;
   set mission_status to continue_unrestricted;
   initialize mission_timer;
   initialize all system objects;

loop: do while mission_status not in
        [mission_complete, abort_to_rendezvous, abort_for_recovery]
   continuously update mission_time;
   if the mission_time = time for some event, then
      perform the event;
   continuously feed waypoints to Guidance subsystem;
   continuously monitor mission progress;
   if an exception occurs, then
      signal possible changes to Guidance, RePlanner;
      access impact of changes;
      reassign mission_status to different category if necessary;
end loop;
```

5.2 Object Representation

Input mission postures are first downloaded from the Mission Planner offboard the vehicle. The input postures are given to the Mission_Interpreter which places a posture into the proper object format and designates the high-level classification of the posture configuration as a transit or specialized mission.

Vehicle internal state is modeled as a set of objects which represents all onboard equipment that need to be monitored. They include:

- Power source: battery_1 and battery_2;
- Control system: rudders, planes, propellers;
- Sonars: forward, left, right, bottom;
- Onboard computers: the Gespac 68030 computer, 386 PC;
- Navigation instrument: gyros, GPS;
- Environment sensors for water temperature and pressure.

Each object representing a piece of equipment has a default guard-line and red-line ranges which, when violated, cause an alarm to be sent to the decision making levels. The guard-line value exists to provide the equipment to degrade more gracefully, by initiating the turn-key operation to energize redundant equipment or power source. The red-line reading (either high or low) indicates the failure point or equipment shutdown limit (naturally, not all equipment or power sources have both high and low limits). The reading of object values is done by a polling rule, monitor_health_continuously which sends for a message at regular intervals from the *Mission Executor*. If the value exceeds the guard-line value, then the *Mission Executor* places the system being monitored in the condition of critical. If the sensor red-line value is exceeded, the equipment is assumed to have failed. In the case of a vehicle control system such as the rudder or diving planes, there is also a *message handler* which checks the response of the system. This often means positional response. For example, if the autopilot generates a command to turn left and the rudder moves in a wrong direction, then the system is assumed to have become critical.

Objects are used to model not only equipments, but also decisions which are maintained for the purposes of later retrieval in reconstructing the mission and in conducting any possible machine learning for AUV II. Maintaining decisions is useful not only for mission documentation, but also in resolving conflicts between states.

5.3 Layering of Rules

The rules in the *Mission Executor* are layered in three levels of reasoning. The lowest-level rules actually carry out the corrective action by ordering *Guidance* to turn, ascend to safe depth. These rules are assumed to be competent operators or controllers. For example, the maneuvering rule abnormal_dive is given the responsibility to order *Guidance* to decrease the speed and ascend to the designated safe depth, bringing the vessel to a safe configuration before it propagates this situation to the intermediate level assessment rule.

The intermediate level consists of assessment rules from functional areas. For example, the Maneuvering_Status_Assessment rule evaluates various types of maneuvering status problems, and tabulates the number of deficiencies. Since the overall mission status is dependent on rapid propagation of changes from the assessment rules, the assessment rules are given a higher salience value to avoid being in conflict with lower level rules. Assigning a higher salience value to the assessment rules gives them adequate priority.

At the highest level, the Overall_Mission_Assessor examines the current status of all functional areas and makes a determination on the state of the vehicle mission. At that point, the overall mission status is changed, if necessary, and the results propagated down to the respective mission abort or mission restricted rules. All of the functional areas have a similar structure. Maneuvering has the added feature of low level assessment rules which examine the obstacle object base to see if the indicated obstacles pose a collision danger.

5.4 Maintaining a Consistent Knowledge Base

As important as sensing data and scheduling actions is the maintenance of consistency in the knowledge base. In a rule-based system this becomes acutely important when the generation of a new action through a control fact is based on some other events. If the events that would cause the action are no longer valid, then it may be the case that the generated control fact is no longer valid. In such a case it will be necessary to remove the fact. This truth maintenance is an integral part of the *Mission Executor*, mostly in

the highest levels. The vehicle's initial state (hence ideal state) rests upon a foundation of all functional areas being operational (this does not mean that all functional areas are devoid of any complications; it only means that the complications will not cause the vehicle to become critical).

Any failure of a particular functional area will cause the mission status to be changed. The *Mission Executor* does this by retracting the mission status of continue_unrestricted and asserting a new status of abort_mission. It results in vehicle recovery or an abort transit to the designated rendezvous. The abort status is one that should remain in effect until the vehicle is recovered. This is so since abort_mission should only take place when all relevant options to continue the mission have been explored and found unexecutable.

However, in the interval between the status change and the actual vehicle recovery, there is the possibility that a functional area becoming critical could later attempt to cause the status of continue_with_restrictions. There is also the possibility that the functional area recovery rules could cause the new state of continue_unrestricted. To counter any possibility that either of these could happen, a truth maintenance feature of *status lock* is incorporated. This causes the mission assessor rule to be removed. Thus, no mission state change can occur and overall mission status becomes "frozen".

5.5 Managing Uncertainty

Uncertainty plays a significant role in the *Mission Executor*. In fact the primary reason for using a forward-chaining rule based tool such as CLIPS is that there is a great deal of uncertainty about the external environment. What is known about the environment can best be classified in heuristics. A specific area of uncertainty that the *Mission Executor* must reason about is the presence of obstacles. Reporting an obstacle at short range automatically generates a command from the *Mission Executor* (an emergency situation) but reporting an obstacle at the limit of the sonar is a different matter. The obstacle is assigned a confidence factor which comes from the *Sonar Processing Suite*. If the confidence factor is high to medium and the obstacle is within the 180 degree arc about the bow of the AUV, then the obstacle is considered to be a collision danger. This will

cause the path to be replanned. The rationale is that the farther away an obstacle is detected, the less radical a turn is necessary, often resulting in less deviation from the original track, saving both mission time and energy consumption.

The confidence factor is checked whenever an obstacle alert flag is sent, be it an update or a new obstacle. This feature helps to maintain the high level configuration while still allowing for the necessary actions of avoiding obstacles and performing routine navigation enroute to the mission origin or designated rendezvous.

5.6 Mission Documentation

There is a vital need for documentation of AUV missions. Most AUV projects have come to rely on some data recorded onboard the AUV. The compilation of data is valuable for several reasons:

- It can be analyzed by human researchers to update and refine the AUV control systems (both hardware and software).
- It can provide an idea of what works with rule-based systems and where failure in reasoning occurs.
- It can be used as a persistent base of knowledge for "training" AUV's in situation assessment.

Documentation already exists within the NPS AUV II baseline system in the form of the *Environmental Database* which contains navigational data and data about obstacles which might be encountered. A mission log is maintained by the *Navigator* module in much the same way as a mission log is kept by a navigator of a maritime vessel. However, in order to adequately study high level control, a mission log must also be kept of high level decisions. It can be regarded as a form of captain's log which records the state of the mission at the highest level and justifications for decisions made. At a standard time interval or whenever the overall mission decision changes, an entry is made to the log.

6 Simulation

A set of five simulation scenarios was designed to determine if the *Mission Executor* can perform the high-level symbolic reasoning that generates the desired decisions for AUV operation. We would also like to know whether the reasoning system could recognize situations and try to approximate real-time constrained decision-making. Navigational waypoints used in the simulation were based on the model of the Naval Postgraduate School pool. An average mission time of two to four minutes was used for each scenario. The simulations were run on a Sun Sparcstation 1.

6.1 Scenarios and Results

Scenario one tested the most basic case, pre-planned mission execution monitoring (that is, waypoint sequencing). The AUV II was given a set of waypoints, each with its specified estimated time of arrival as a constraint. At the third waypoint, the vehicle missed its time constraint by 47 seconds, longer than the predefined tolerance level of 40 seconds, and enough to cause the Waypoint_DistanceTime_Check rule to alert the Navigation_Assessment rule. This resulted in a command to increase speed, but no change in navigation status, since one violation of this kind is not considered critical (the Navigation_Assessment rule employs a heuristic for replanning if four navigation problems have occurred). In this scenario, it took 0.28 seconds for the *Mission Executor* to recognize the large navigational discrepancy in time (basically the work of Waypoint_DistanceTime_Check rule). It took the Navigation Assessment rule 0.16 seconds to determine that there was no need to change the status. The overall elapsed mission time was 3 minutes 21 seconds with 16936 rules fired.

Scenario two tested the ability of the *Mission Executor* to recognize an untenable obstacle avoidance situation. Both short-range obstacles and long-range obstacles were tested. The first recognition of an obstacle close aboard led to an ascent to safe-depth. This also tested a rule recognizing possible shoaling or grounding of the vessel. The emergency avoidance maneuver rule began its time check of the avoidance maneuver. An obstacle detected at long range led to assessment of the obstacle as threatening to

the AUV. The overall maneuvering status was changed to continue_with_restrictions. At one point enough obstacles had accumulated to cause the Collective_Obstacle_Assessment rule to characterize the situation as involving a critical number of obstacles (the criterion is four separate encounters). Later the same rule determined that the critical point had been breached by the accumulation of too many obstacles along the track (the heuristic is based on the reasoning that too many obstacles will cause too many time-consuming avoidance maneuvers). The Maneuvering_Status_Assessment rule determined that this was a functional area failure, resulting in an abort-mission situation. From recognition of the critical point at 50.45 seconds into the mission, it took approximately 6.55 seconds to recognize the undesirable situation. The change in the maneuvering status and subsequent overall assessment of the mission resulted in a time of propagation of 0.14 seconds. The overall elapsed mission time was 1 minute 57 seconds with 8610 rules fired.

Scenario three involved a vehicle control system failure. After passing several way-points, the AUV experienced an electrical failure of diving planes which triggered the rule Maneuvering_Equipment_Failure first. The Control_System_Failure rule fired shortly after that, leading to the overall mission assessment that this was an abort situation. From the instantiation of the triggering event until the time it was recognized as an abort situation was an interval of 0.24 seconds. Propagation of the maneuvering status or equipment status to the Overall_Mission_Assessor was difficult to absolutely determine because of the fact that both maneuvering assessment and equipment status assessment fired. Either one could have caused the overall mission status to change. Because of the high salience of both rules, activation of the Overall_Mission_Assessor occurred only 0.17 seconds after the Equipment_Status_Assessment rule fired. The overall elapsed mission time was 41 seconds with 2500 rules fired.

Scenario four evaluated both obstacle avoidance and environmental phenomena. Only two obstacle encounters were realized, resulting in only minor deviations to the planned navigational track. A significant environmental phenomenon was simulated by having readings in all three environmental sensors exceeding allowable limits. This resulted in a

mission abort. After the time of the triggering event, it took the Overall_Mission_Assessor 0.56 seconds to recognize that it was an abort situation. The overall elapsed mission time was 2 minutes 11 seconds with 11200 rules fired.

Scenario five tested multiple equipment failures. The AUV passed through several waypoints missing only one time constraint. A sonar failure (forward sonar) led to a reduction in the overall mission status to continue_with_restrictions as the sonar went to a critical state. A second sonar (right port sonar) led to a reinforcement of that state. Failure of the rudder finally led to the AUV surfacing and energizing its transponder. From the triggering event until the decision to abort, 0.47 seconds elapsed. The overall elapsed mission time was 1 minute 15 seconds with 5456 rules fired.

6.2 Evaluation

Comparison of the results reveals that propagation of status from the functional area assessors to the Overall_Mission_Assessor will probably meet real-time constraints in the relatively slow-moving environment of the AUV in its testing facility. The true time dependency does appear to be in the low-level action or assessment rules. Situation recognition depends on good heuristics.

The use of a layered situation-based reasoning system appears to be sound. By using an intermediate level assessment rule, the desired rapid reaction can be taken at the low-level and the assessment of functional state can proceed at the same time. Thus, there need not be a salience assigned to every level. Refinement of heuristics will certainly be necessary to further optimize the rule base.

7 Conclusions

We have designed and developed a prototype *Mission Executor* for the AUV II, the autonomous underwater vehicle that has been fabricated at the Naval Postgraduate School. This not only provides a high level, intelligent software control for the AUV II, but also allows for the encoding of domain specific heuristics into a rule base. We believe that the essence of the expert knowledge of a submarine captain regarding maneuvering, avoid-

ance of uncharted obstacles, waypoint navigation, and reaction to emergencies has been adequately captured and implemented in our prototype system. We have also identified the major data flows between the AUV II software components and defined the interrupt commands for the *Guidance* subsystem.

While it is impossible to test all possible scenarios, the testing and debugging of the *Mission Executor*, implemented in CLIPS version 5.0, illustrates its rapid prototyping capabilities and the great utility of objects to represent the onboard systems and decisions. Rules for newly-envisioned situations can be added with relative ease. Thus, the prototype is easily extensible. This also means that current rules can be further refined with more test runs in submarine maneuvering.

Through simulations, we have demonstrated that the *Mission Executor* is able to react to adverse situations in a timely manner and to make informed decisions for AUV maneuvering. We plan to install it onto the AUV II when the current development of other software modules is completed.

References

[1] J. G. Bellingham and T. R. Consi. State configured layered control. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, pages 75–80, Monterey, California, October 23-26 1990.

[2] D. R. Blidberg, S. Chappell, J. Jalbert, R. Turner, G. Sedor, and P. Eaton. The EAVE AUV program at the marine systems engineering laboratory. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, pages 33–42, Monterey, California, October 23-26 1990.

[3] R. A. Brooks. A layered intelligent control system for a mobile robot. In Fanguera and Girald, editors, *Robotics Research*, pages 365–372. MIT Press, Cambridge, 1986.

[4] T. Doi and S. Nonose. Total system of advanced subsea robot. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, pages 123–128, Washington DC, 4-6 June 1990. IEEE Oceanic Engineering Society.

[5] C. A. Floyd. Design and implementation of a collision avoidance system for the NPS Autonomous Underwater Vehicle (AUV II) utilizing ultrasonic sensors. Master's thesis, Naval Postgraduate School, Monterey California, September 1991.

[6] J. Giarratano. *CLIPS User's Guide: Volume 1 Rules and Volume 2 Objects.* Software Technology Branch, Lyndon B. Johnson Space Center, Houston, Texas, 5.0 edition, January 1991.

[7] A. J. Healey, R. B. McGhee, R. Cristi, F. A. Papoulias, S. H. Kwak, Y. Kanayama, and Y. Lee. Mission planning, execution and data analysis for the NPS AUV II autonomous underwater vehicle. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, pages 177–186, Monterey California, October 1990.

[8] Isik-90. Pilot level of a hierarchical controller for an unmanned mobile robot. *IEEE Journal of Robotics and Automation*, 4(3), Winter 1989.

[9] M. Iwasaki, J. Akizono, T. Nemoto, and O. Asakura. Field test of aquatic walking robot for undersea inspection. In *Proceedings of the First IARP Workshop on Mobile Robots for Subsea Environments*, pages 109–112, Monterey California, October 1990.

[10] R. E. Korf. Real-time heuristic search: New results. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 139–144, Saint Paul, Minnesota, August 21-26 1988. American Association of Artificial Intelligence.

[11] S. Kwak, S. Ong, , and R. B. McGhee. A mission planning expert system for an autonomous underwater vehicle. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, pages 123–128, Washington DC, 4-6 June 1990. IEEE Oceanic Engineering Society.

[12] Y. Lee and J. Bonsignore. Underwater multi-dimensional path planning. In *Proceedings of the Tenth Annual National Conference on Ada Technology*, Arlington, Virginia, 24-27 February 1992.

[13] McCartney-Collar-90. Autonomous submersibles – instrument platforms of the future. *Underwater Technology*, 15(4):19, 1990.

[14] G. R. Meijer. Exception handling system for autonomous robots based on PES. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, Amsterdam, the Netherlands, December 1989.

[15] N. Polmar. Robot submarines. *U. S. Naval Institute Proceedings*, 118(9):1, September 1991.

[16] O. J. Rodseth. Object-oriented software for AUV control. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, pages 15–24, Monterey, California, October 23-26 1990.

[17] R. B. Schudy and C. N. Duarte. Advanced autonomous underwater vehicle software architecture. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, Washington, DC, June 5-6 1990.

[18] A. S. Westneat and S. L. Clearwaters. A generalized alternative contingency matrix for the autonomous untethered vehicle (AUV). In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, Washington, DC, June 5-6 1990.

[19] X. Zheng, E. Jackson, , and M. Kao. Object-oriented software architecture for mission-configurable robots. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, Monterey, California, October 23-26 1990.

Distribution List

Defense Technical Information Center Cameron Station Alexandria, VA 22304–6145	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943	2
Chairman, Computer Science Department Code CS Naval Postgraduate School Monterey, CA 93943	1
Prof. Yuh-jeng Lee Code CS/Le Naval Postgraduate School Monterey, CA 93943	30